PATENT
5760-09200
VRTS-0220

"EXPRESS MAIL" MAILING LABEL NUMBER
EV318248122US
DATE OF DEPOSIT 11-26-03
I HEREBY CERTIFY THAT THIS PAPER OR FEE IS BEING DEPOSITED WITH THE UNITED STATES POSTAL SERVICE "EXPRESS MAIL POST OFFICE TO ADDRESSEE" SERVICE UNDER 37 C.F.R. § 1.10 ON THE DATE INDICATED ABOVE AND IS ADDRESSED TO THE COMMISSIONER FOR PATENTS, P.O. BOX 1450, ALEXANDRIA, VA 22313-1450.

[illegible]
Derrick Brown

SYSTEM AND METHOD FOR EMULATING OPERATING SYSTEM METADATA TO PROVIDE CROSS-PLATFORM ACCESS TO STORAGE VOLUMES

By:

Ronald S. Karr

Oleg Kiselev

Alex Miroschnichenko

B. Noël Kivlin/RPH
Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8800

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] This invention is related to the field of storage management and, more particularly, to the integration of off-host storage virtualization with hosts in a heterogeneous computing environment.

Description of the Related Art

[0002] Modern enterprise computing environments are characterized by pressures for continuous availability and growth. These factors are driving the adoption of new approaches to computing and storage. Enterprise computing environments are increasingly using computer clusters, Storage Area Networks (SANs), and other centralized storage mechanisms to simplify storage, improve availability, and handle escalating demands for data and applications.

[0003] Clustering may be defined as the use of multiple computers (e.g., PCs or UNIX workstations), multiple storage devices, and redundant interconnections to form what appears to external users as a single and highly available system. Clustering may be used for load balancing and parallel processing as well as for high availability.

[0004] The storage area network (SAN) model places storage on its own dedicated network, removing data storage from the main user network. This dedicated network most commonly uses Fibre Channel technology as a versatile, high-speed transport. The SAN includes one or more hosts that provide a point of interface with LAN users, as well as (in the case of large SANs) one or more fabric switches, SAN hubs and other devices to accommodate a large number of storage devices. The hardware (e.g. fabric switches, hubs, bridges, routers, cables, etc.) that connects workstations and servers to storage devices in a SAN is referred to as a "fabric." The SAN fabric may enable server-to-storage device connectivity through Fibre Channel switching technology to a wide range of servers and storage devices.

[0005] The versatility of the SAN model enables organizations to perform tasks that were previously difficult to implement, such as LAN-free and server-free tape backup, storage leasing, and full-motion video services. SAN deployment promises numerous advantages, including cost management through storage consolidation, higher availability of data, better performance and seamless management of online and offline data. In addition, the LAN is relieved of the overhead of disk access and tape backup, data availability becomes less server-dependent, and downtime incurred by service and maintenance tasks affects more granular portions of the available storage system.

[0006] A block server enables a computer system to take its storage and "serve" that storage onto a SAN (e.g., as virtual SCSI disks). A block server may also be referred to herein as a "block-server appliance" or "appliance." The block server and its virtual disks may have many advantages for ease of management and consolidation of storage. For example, a block server may provide the ability to easily allocate and reallocate storage on a SAN: the right amount of storage to the right computer system at the right time. This functionality may be thought of as "repurposing" the storage from the application server perspective. A block server may also provide the ability to consolidate storage behind the SAN (i.e., managing the storage pool). For example, this consolidation can be employed with pre-existing fibre channel or SCSI storage not on a SAN. One could then "repurpose" non-SAN storage and move it into a managed SAN environment.

[0007] A block server enables a computer system to take its storage and "serve" that storage onto a SAN (e.g., as virtual SCSI disks). As used herein, a "block server" comprises a hardware or software entity that provides a collection of linearly addressed blocks of uniform size that can be read or written. A block server may also be referred to herein as a "block device, "block-server appliance," or "appliance." The block server and its virtual disks may have many advantages for ease of management and consolidation of storage. For example, a block server may provide the ability to easily allocate and reallocate storage on a SAN: the right amount of storage to the right computer system at the right time. This functionality may be thought of as "repurposing" the storage from the application server perspective. A block server may also provide the ability to consolidate storage behind the SAN (i.e., managing the storage pool). For example, this consolidation can be employed with pre-existing fibre channel or SCSI storage not on a SAN. One could then "repurpose" non-SAN storage and move it into a managed SAN environment.

[0008] A block device differs from a file in that it does not require use of a file system, and is typically less dynamic. A block device presented by an operating system presents relatively few primitives: open, close, read, write, plus a few miscellaneous control and query primitives. File systems provide a richer set of primitives, including support for creating and removing files, appending to files, creating and removing directories, etc. Typical interfaces to block devices also allow for higher raw throughput and greater concurrency than typical interfaces to single files of a file system. Block devices residing on hardware devices typically present some form of SCSI interface, though other interfaces are possible.

[0009] A basic block device comprises a simple array of blocks. The prototypical block device is a single disk drive presenting all of its sectors as an indexed array blocks. Disk arrays and volume managers introduce virtualization of blocks, creating some number of virtual block devices. In block virtualization, one or more layers of software and/or hardware rearrange blocks from one or more disks, add various kinds of functions, and present the aggregation to a layer above as if it were essentially a collection of basic disk drives (i.e., presenting the more complex structure as if it were simple arrays of blocks). Block virtualization can add aggregation (striping and spanning), mirroring and other forms of redundancy, some performance optimizations, snapshots, replication, and various capabilities for online reorganization. Block virtualization provides all these capabilities without affecting the structure of the file systems and applications that use them.

[0010] As used herein, a "logical volume" comprises a virtualized block device that is presented directly for use by a file system, database, or other applications that can directly use block devices. This differs from block devices implemented in hardware devices, or below system disk drivers, in that those devices do not present a direct system device that can be opened for direct use. Instead, a system-dependent disk driver is typically used to access the device. The disk driver is generally unaware of the hardware virtualization, but adds some limited virtualization of its own (often just segmenting in the form of partitions). The disk driver also forms an abstraction barrier that makes it more difficult for applications and file systems to cooperate with the underlying virtualization in advanced ways.

[0011] As used herein, a "logical or physical disk [drive]" (also referred to as a "physical volume") comprises a disk drive (physical) or a device (logical) presented by a hardware block virtualizer to look like a disk drive. Disk arrays present logical disks. Virtualizing host bus adapters and many virtualizing switches also present logical disks. Upper layers of virtualization typically run on top of logical disks.

[0012] Distributed block virtualization may distribute a description of how a virtual block device (for example, a logical volume or a virtual Logical Unit) relates to underlying storage, as well as how distributed block virtualization components might relate in order to accomplish certain functions. As used herein, "distributed [block] virtualization" typically refers to what is commonly called "out-of-band" virtualization. Block virtualization is basically the concept of defining a more complex structure between a consumer of a block device and the underlying block storage. The block device presented is often called a logical volume. Distributed block virtualization somehow communicates that structure between systems either so that several systems can share parts of underlying storage that is managed above by a virtualizer, and so that the implementation of some of the block virtualization operations can be distributed and coordinated.

[0013] Traditionally, block storage architectures included the following layers in the I/O stack of the host computer system: file system, swap, or database; logical partitioning in a disk driver; and an interconnect-dependent I/O driver. A storage device (or other storage subsystem outside of the host) typically included an interconnect-dependent target firmware and one or more disks. Logical partitioning in the disk driver was developed as a way of subdividing the space from disk drives so that several smaller file systems (or a raw swap device) could be created on a single disk drive.

[0014] Operating systems differ in how they perform logical partitioning. Most operating systems (e.g., Solaris, Windows, and Linux) use the simple partitioning described above. Some UNIX-based operating systems from HP and IBM, however, do not implement simple logical partitioning in their disk drivers; instead, these operating systems include a virtualization layer that defines logical volumes that can subdivide or span disks in more flexible ways. These logical volumes may be referred to as "host-virtual objects." The virtualization layer used with host-virtual objects may employ complex on-disk metadata, potentially spread across several disks and potentially including additional metadata stored elsewhere, to define a virtual structure. For both simple partitions and host-virtual objects, the storage management function of dividing storage is implemented in host software, and underlying disk storage subsystems are expected basically to supply raw storage containers which are not used directly by file systems or applications.

[0015] Evolution in this storage management structure has occurred through increased complexity of each layer, or by introducing intermediate layers that emulate the relationships below and above and leaving surrounding layers unchanged. Most commonly, block virtualization layers have been added in at various places in the I/O stack. Block virtualization layers have been added between the file system and partitioning disk driver, between the target firmware and the disk (e.g., disk arrays), and between the interconnect driver in the system and the physical bus (e.g., virtualizing host bus adapters). More recently, there has been a trend toward adding virtualization into the physical interconnect itself (e.g., virtualizing SAN switches and block-server appliances).

[0016] If block virtualization is implemented outside of the host, and if preservation of the structure of the storage stack is required, then external virtualization may be used to present standard SCSI LUNs (with optional extensions) to the host. A Logical Unit is a block device with an interface presented on a storage bus or a storage network as described, for example, by the SCSI standard. LUN (logical unit number) is a unique identifier used on a SCSI bus to distinguish between devices that share the same bus. These numbers are used to address Logical Units over one or another interface accessed by some connected host or device. As used herein, the term "storage device" is intended to include SCSI Logical Unit devices as well as other suitable hardware devices.

[0017] In the process of presenting the blocks of the block-virtual volumes directly to file systems or block-consuming applications, several problems may be encountered. First, without a consistent name for a block device, applications and file systems may not know what they are accessing, and the administrator of an environment may not understand the relationship between the device that the file system is using and the virtual volume defined by the block virtualization.

[0018] Second, any partitioning or virtualization schemes that are specific to a particular operating system may interfere with cross-platform (e.g., cross-operating-system) access to external storage volumes. Modern enterprise computing environments may include computer systems and storage devices from many different vendors. The computer systems may be operating under different operating systems, each of which may use its own file system. Different file systems may each feature their own proprietary sets of metadata relating to their underlying data objects. For these reasons and more, offering uniform access to storage devices in a heterogeneous computing environment can be problematic.

SUMMARY OF THE INVENTION

[0019] Various embodiments of a method and system for emulating operating system metadata to provide cross-platform access to storage volumes are disclosed. In one embodiment, the method may include generating operating system metadata for a storage device, wherein the operating system metadata emulates a storage volume hosted under a first operating system. The method may further include making the operating system metadata available to a host computer system, wherein the host computer system runs the first operating system. The operating system metadata enables the host computer system to recognize the storage device as the storage volume hosted under the first operating system.

BRIEF DESCRIPTION OF THE DRAWINGS

[0020] Figure 1 illustrates an exemplary enterprise computing system, including a storage area network, with which embodiments of a system and method for providing cross-platform access to storage volumes may be implemented.

[0021] Figure 2 illustrates an exemplary computer system with which embodiments of a system and method for providing cross-platform access to storage volumes may be implemented.

[0022] Figure 3 illustrates an architecture of software and/or hardware components for providing cross-platform access to storage volumes according to one embodiment.

[0023] Figure 4 illustrates examples of emulated storage volumes according to one embodiment.

[0024] Figure 5 is a flowchart which illustrates a method for emulating operating system metadata to provide cross-platform access to storage volumes according to one embodiment.

[0025] While the invention is described herein by way of example for several embodiments and illustrative drawings, those skilled in the art will recognize that the invention is not limited to the embodiments or drawings described. It should be understood, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents and alternatives falling within the spirit and scope of the present invention as defined by the appended claims. As used throughout this application, the word "may" is used in a permissive sense (i.e., meaning having the potential to), rather than the mandatory sense (i.e., meaning must). Similarly, the words “include”, “including”, and “includes” mean including, but not limited to.

DETAILED DESCRIPTION OF EMBODIMENTS

[0026] Various embodiments of the system and method disclosed herein provide cross-platform accessibility of logical storage volumes that are encapsulated within and exported by various storage devices or systems. Typically, the storage devices or systems will export block devices using basic SCSI or other appropriate protocols in a way that is picked up by regular system disk drivers. Using the system and method disclosed herein, a logical volume which is defined externally to one or more hosts may be made available as that logical volume to the one or more hosts, even if those hosts use different operating systems and different kinds of disk drivers for accessing external storage. The system and method may be used with various kinds of storage networks and intelligent disk arrays that are locally attached to systems using various kinds of I/O buses. The system and method described herein may be referred to as "volume tunneling."

[0027] Figure 1 illustrates an exemplary Storage Area Network (SAN) environment with which embodiments of a system and method for providing cross-platform access to storage volumes may be implemented. For one embodiment, the SAN may be described as a high-speed, special-purpose network that interconnects one or more storage devices 104 (e.g. storage devices 104A, 104B, and 104C) with one or more associated host systems or servers 102 on behalf of a larger network of users. This dedicated network may employ Fibre Channel technology. A SAN may be part of the overall network of computing resources for an enterprise or other entity. The one or more servers 102 and one or more storage devices 104 (e.g. storage devices 104A, 104B, and 104C) may be coupled via a fabric 100. One or more client systems 106 may access the SAN by accessing one or more of the servers 102 via a network 108. The client systems 106 may communicate with the server 102 to access data on the storage devices 104A, 104B, and 104C which are managed by server 102. The client systems 106 may comprise server systems which manage a different set of storage devices. The client systems 106 may therefore also act as servers and may be coupled to other storage devices (not shown) through the fabric 100.

[0028] Network 108 may include wired or wireless communications connections separate from the Fibre Channel network. For example, network 108 is representative of any local area network (LAN) such as an intranet or any wide area network (WAN) such as the Internet. Network 108 may use a variety of wired or wireless connection mediums. For example, wired mediums may include: a modem connected to plain old telephone service (POTS), Ethernet, and fiber channel. Wireless connection mediums include a satellite link, a modem link through a cellular service or a wireless link such as Wi-Fi™, for example.

[0029] Storage devices may include any of one or more types of storage devices including, but not limited to, storage systems such as RAID (Redundant Array of Independent Disks) systems, disk arrays, JBODs (Just a Bunch Of Disks, used to refer to disks that are not configured according to RAID), tape devices, and optical storage devices. These devices may be products of any of a number of vendors including, but not limited to, Compaq, EMC, and Hitachi. Clients 106 and server 102 may run any of a variety of operating systems, including, but not limited to, Solaris 2.6, 7 or 8, Microsoft Windows NT 4.0 (Server and Enterprise Server), Microsoft Windows 2000 (Server, Advanced Server and Datacenter Editions), and various versions of HP-UX. Each server 102 may be connected to the fabric 100 via one or more Host Bus Adapters (HBAs).

[0030] The hardware that connects servers 102 to storage devices 104 in a SAN may be referred to as a fabric 100. The SAN fabric 100 enables server-to-storage device connectivity through Fibre Channel switching technology. The SAN fabric 100 hardware may include one or more switches (also referred to as fabric switches), bridges, hubs, or other devices such as routers, as well as the interconnecting cables (for Fibre Channel SANs, fibre optic cables). SAN fabric 100 may include one or more distinct device interconnection structures (e.g. Fibre Channel Arbitrated Loops, Fibre Channel Fabrics, etc.) that collectively form the SAN fabric 100.

[0031] In one embodiment, a SAN-aware file system may use the Network File System (NFS) protocol in providing access to shared files on the SAN. Using NFS, each server 102 may include a logical hierarchy of files (i.e. a directory tree) physically stored on one or more of storage devices 104 and accessible by the client systems 106 through the server 102. These hierarchies of files, or portions or sub-trees of the hierarchies of files, may be referred to herein as "file systems." In one embodiment, the SAN components may be organized into one or more clusters to provide high availability, load balancing and/or parallel processing. For example, in Figure 1, server 102 and clients 106A and 106B may be in a cluster.

[0032] In traditional storage architecture, each server is privately connected to one or more storage devices using SCSI or other storage interconnect technology. If a server is functioning as a file server, it can give other servers (its clients) on the network access to its locally attached files through the local area network. With a storage area network, storage devices are consolidated on their own high-speed network using a shared SCSI bus and/or a fibre channel switch/hub. A SAN is a logical place to host files that may be shared between multiple systems.

[0033] A shared storage environment is one in which multiple servers may access the same set of data. A challenge with this architecture is the maintenance of consistency between file data and file system data. A common architecture for sharing file-based storage is the file server architecture (e.g., the SAN environment illustrated in Figure 1). In the file server architecture, one or more servers are connected to a large amount of storage (either attached locally or in a SAN) and provide other systems access to this storage.

[0034] Figure 2 illustrates an exemplary computer system 106 with which embodiments of a system and method for providing cross-platform access to storage volumes may be implemented. Generally speaking, the client computer system 106 may include various hardware and software components. In the illustrated embodiment, the client computer system 106 includes a processor 220A coupled to a memory 230 which is in turn coupled to storage 240. In addition, processor 220A is coupled to a network interface 260. The client computer system 106 may be connected to a network such as network 108 via a network connection 275. Further, the client computer system 106 includes operating system software 130. The operating system software 130 is executable by processor 220A out of memory 230A. The operating system software 130 may include an I/O or storage stack 250. Typically, the I/O stack 250 may include one or more file systems, volume managers, and device drivers.

[0035] Processor 220A may be configured to execute instructions and to operate on data stored within memory 230A. In one embodiment, processor 220A may operate in conjunction with memory 230A in a paged mode, such that frequently used pages of memory may be paged in and out of memory 230A from storage 240 according to conventional techniques. It is noted that processor 220A is representative of any type of processor. For example, in one embodiment, processor 220A may be compatible with the x86 architecture, while in another embodiment processor 220A may be compatible with the SPARC™ family of processors.

[0036] Memory 230A is configured to store instructions and data. In one embodiment, memory 230A may be implemented in various forms of random access memory (RAM) such as dynamic RAM (DRAM) or synchronous DRAM (SDRAM). However, it is contemplated that other embodiments may me implemented using other types of suitable memory.

[0037] Storage 240 is configured to store instructions and data. Storage 240 may be an example of any type of mass storage device or system. For example, in one embodiment, storage 240 may be implemented as one or more hard disks configured independently or as a disk storage system. In one embodiment, the disk storage system may be an example of a redundant array of inexpensive disks (RAID) system. In an alternative embodiment, the disk storage system may be a disk array, or Just a Bunch Of Disks (JBOD), (used to refer to disks that are not configured according to RAID). In yet other embodiments, storage 240 may include tape drives, optical storage devices or RAM disks, for example.

[0038] Network interface 260 may implement functionality to connect the client computer system 106 to a network such as network 108 via network connection 275. For example, network interconnect 260 may include a hardware layer and a software layer which controls the hardware layer. The software may be executable by processor 220A out of memory 230A to implement various network protocols such as TCP/IP and hypertext transport protocol (HTTP), for example.

[0039] While Figures 1 and 2 illustrate typical computer system and network storage architectures in which embodiments of the system and method for providing cross-platform access to storage volumes may be implemented, embodiments may be implemented in other computer and network storage architectures including other SAN architectures.

[0040] Figure 3 illustrates an architecture of software and/or hardware components for providing cross-platform access to storage volumes according to one embodiment. Embodiments of a server system 102 may be similar to the client system 106 illustrated in Figure 2. The server system 102 may include a processor 220B coupled to a memory 230B which is in turn coupled to optional storage. The server computer system 102 may be connected to a network via a network connection. In one embodiment, the server computer system 102 may include operating system and/or server software which is executable by processor 220B out of memory 230B. Using the operating system software and/or server software, the server system 102 may "own" or manage data objects (e.g., files) on one or more storage devices 104. In one embodiment, the server system 102 may execute file server software to control access to the data it owns. In another embodiment, the server functionality may be implemented in hardware and/or firmware.

[0041] In one embodiment, the server 102 may take the form of a block-server appliance (e.g., on a SAN). The server 102 may be referred to herein as a "storage virtualization controller" or "storage virtualizer." As used herein, a storage virtualization controller is a computer system or other apparatus which is operable to emulate operating-system-specific storage metadata to enable cross-platform availability of managed storage devices 104.

[0042] Storage devices are usually designed to provide data to servers 102 using one of two methods, either block-level or file-level access. Block I/O is typically the I/O which is tied directly to the disks 104 and used when direct and fast access to the physical drives themselves is required by the client application. The client 106 may request the data from the block server 102 using the starting location of the data blocks and the number of blocks to be transferred. In one embodiment, the operating system 130 on the client 106 may provide a file system to help translate the block locations into file locations.

[0043] The memory 230B of the server 102 may store instructions which are executable by the processor 220B to implement the method and system for providing cross-platform access to storage as disclosed herein. The instructions may include multiple components or layers. For example, device drivers 256 may be used for I/O between the server and the storage devices 104. Components such as an emulation layer 252 and/or storage virtualizer 254 may be used to provide uniform access to data for heterogeneous clients, as will be discussed in greater detail below.

[0044] The connection 109 between the block server 102 and the client 106 may comprise a channel protocol such as SCSI or fibre channel which may also be connected directly to disks or disk controllers (e.g., RAID controllers). Since SCSI and fibre channel are protocols having relatively low overhead, they may provide a low latency connection with high performance.

[0045] Figure 4 illustrates examples of emulated storage volumes according to one embodiment. One or more storage devices (e.g., devices 104A, 104B, and 104C) may include one or more storage volumes (e.g., volumes 105A, 105B, 105C, 105D, and 105E). Because a volume is an abstract logical entity, it can start and end anywhere on a physical disk or in a partition and be composed of space on physical disks on different devices, using a variety of organizational styles, including simple aggregation, mirroring, striping, and RAID-5. Physical disk space can also be used within a volume or a collection of volumes for storing additional information, such as update logs, configuration information, or various kinds of tracking structures. Therefore, a single storage device 104 may comprise more than one logical volume 105, and a single volume 105 may span two or more physical devices 104.

[0046] When reading data from storage through its own I/O stack 250, a client computer system 106 may expect the data to be formatted in a certain way. In other words, each client 106 may expect the data to be associated with metadata that is specific to the particular operating system 130 executed by the client 106. As used herein, an "operating system" ("OS") is a set of program instructions which, when executed, provide basic functions, such as data input/output ("I/O"), for a computer system. As used herein, "metadata" generally refers to data that identifies the formatting, structure, origin, or other attributes of other data, for example, as needed to satisfy the native requirements of a particular operating's block storage I/O subsystems.

[0047] Using the method and system for emulating operating system metadata disclosed herein, OS-specific emulated storage volumes 107 may be created from the logical storage volumes 105 on the physical devices 104. In one embodiment, the Logical Unit presented for the external volume may be "decorated" with additional metadata which is supplied before and/or after the actual volume contents. This metadata enables the operating system (OS) on the host to recognize that the storage device contains a partition or other virtual structure (e.g., a host-virtual object) that happens to map to the actual contents of the volume. By generating OS-specific metadata for a storage volume 105, an OS-specific emulated volume 107 may be created that satisfies the I/O expectations of a client running the specific OS.

[0048] Different operating systems may have different requirements in accessing volumes on storage devices, and thus different sets of operating system metadata may be used for different operating systems. Therefore, the storage virtualizer may be configured to provide different sets of operating system metadata for different storage devices. If a storage device can be shared (either serially or concurrently) by two different types of operating systems, then the storage virtualizer may be configured to provide different sets of operating system metadata for the same storage device.

[0049] In one embodiment, multiple emulated volumes 107 may therefore be generated from a single volume 105. For example, OS-specific volumes 107A, 107B, and 107C may emulate storage volume 105A for particular respective operating systems. Emulated volume 107D may emulate volume 105D for the Solaris operating system. Emulated volumes 107E and 107F may emulate volume 105E for the Windows NT and HP-UX operating systems, respectively.

[0050] Figure 5 is a flowchart which illustrates a method for emulating operating system metadata to provide cross-platform access to storage volumes according to one embodiment. The method may be performed in whole or part by a storage virtualization controller or server 102 (as illustrated in Figure 3).

[0051] The actual blocks of an external volume (e.g., a volume defined in an external virtualizer) are made available as a range of blocks within the storage device. In 301, additional blocks of the storage device (typically before and/or after the volume) are created in order to satisfy the OS-specific needs of a partitioning disk driver or host-based virtualization layer in the host. These additional blocks contain operating system metadata to satisfy the OS-specific needs of the host. When associated with a storage volume 105, the operating system metadata may effectively emulate the storage volume as it would look if hosted under a particular operating system (e.g., Solaris, Windows NT, HP-UX, etc.), thereby generating an emulated storage volume. An "emulated storage volume" refers to a storage volume which imitates a storage volume hosted under a particular operating system. As used herein, the phrase "hosted under an operating system" refers to the host computer running the particular operating system and/or the volume being formatted using the particular operating system.

[0052] The contents of the metadata blocks may be sent to the host in 303. Using the metadata, the host (e.g., a disk driver or host-based virtualization layer) is able to recognize that the storage device contains an addressable object (e.g., a partition or host-virtual object) whose offset and length correspond to the range of blocks that map the actual external volume within the storage device.

[0053] In one embodiment, a driver on the host may use a form of operating system metadata to locate the actual volume contents within the presented storage device. The driver may be located in a layer above the basic disk driver. In this embodiment, the contents of the metadata blocks may satisfy the needs of the OS disk driver stack. Furthermore, the contents of the metadata blocks may allow the storage device to contain unmolested blocks that map to the external volume and that can be located within the storage device by the special driver. In one embodiment, a structure similar to a partition table may identify a partition that maps a volume, in which case the OS metadata may be used to locate the volume. In an alternative embodiment, additional data may identify a location within the emulated volume where a layered driver could find the volume. The additional data may be in-band within the emulated data, from in-band SCSI data (such as mode pages), or from an out-of-band source (e.g., communicated over a network from another computer or over a network other than the storage network from the virtualizing device).

[0054] In one embodiment, the metadata blocks may either be stored and logically concatenated with the volume. In one embodiment, the metadata blocks can be generated on the fly in response to requests to read blocks from the storage device. Because the contents of the metadata blocks are predictable (if the software and OS on the host are known), it may be more efficient to generate the operating system metadata on the fly. Nevertheless, storing the metadata blocks may allow the format of the blocks to be programmed externally by an agent outside of the storage virtualizer. This external agent may understand the format of an arbitrary operating system, including one that was not known when the storage virtualization software was written and deployed.

[0055] In one embodiment, the method shown in Figure 5 may be performed in response to an I/O request from the host computer system. A data request may be received from a client computer system 106. The data request may comprise a request to access a set of data which is stored on one or more storage volumes 105 and whose access is controlled by the server 102. For example, the client system 106 may send the data request in response to a request from the client's application software to read, write, delete, change ownership or security attributes, or perform another I/O function on a data object (e.g., a file) that is managed by the server 102. A client's operating system software 130, including an I/O stack 250, may handle the sending and receiving of data regarding the data request and any responses from the server 102. A data request may include a file identifier (e.g., a file name, file handle, or file ID number) to identify the requested file and any additional information which is useful for performing the desired operation. For example, a "read" request may include the file handle, an offset into the file, a length for the read, and a destination address or buffer of the read.

[0056] The operating system metadata may comprise different information for different operating systems. For example, the operating system metadata may comprise information which is located at the beginning of the emulated storage volume and information which is located at the end of the emulated storage volume. The operating system metadata may identify the set of data as being stored at a particular location on the emulated storage volume. The operating system metadata may comprise an identifier for the emulated storage volume. For example, a Solaris volume typically comprises an identifying VTOC (virtual table of contents) at a particular location on the volume, usually in partition 1 with a copy in the last two cylinders. The operating system metadata may comprise a cylinder alignment and/or cylinder size for the emulated storage volume. The operating system metadata may include appropriate OS-specific boot code if the volume is intended to be bootable. Typically, a boot program does little more than determine where to find the real operating system and then transfer control to that address.

[0057] In one embodiment, a Windows NT or Windows 2000 volume may include the following characteristics: a "magic number" (i.e., numbers that the OS expects to see, usually at or near the start of the disk); a Master Boot Record (MBR) comprising all 512 bytes of sector zero; a fixed size and position on a disk or over a set of disks; a particular cylinder alignment organized as a number of 512-byte blocks that can be read or written to by the OS; includes one or more subdisks (Windows 2000) as a group of subdisks called plexes; and may or may not include a file system. These characteristics may be emulated by operating system metadata generated in 305.

[0058] In one embodiment, a Solaris volume includes a VTOC (virtual table of contents), typically in partition 1. The VTOC may be emulated by operating system metadata generated in 305. The VTOC may include a layout version identifier, a volume name, the sector size (in bytes) of the volume, the number of partitions in the volume, the free space on the volume, partition headers (each including an ID tag, permission flags, the partition's beginning sector number, and the number of blocks in the partition), and other suitable information.

[0059] System-to-system variations in operating system metadata may be configured in a variety of ways. First, a storage device may be configured to respond with a particular metadata format to particular specified hosts (e.g., particular specified I/O controllers). Second, a management infrastructure layer may communicate with host-based software, determine the necessary metadata format to use for a particular host, and communicate the information to the storage virtualizer. Third, an extended I/O request (e.g., a vendorunique SCSI request) may be used to directly inform the storage virtualizer of the required metadata format.

[0060] In one embodiment, a layered driver that locates the external volume within the storage device may be used instead of system-to-system variations in operating system metadata. In a cross-platform data sharing environment, it may be possible to define a consistent format that satisfies the needs of disk drivers on various operating systems (e.g., Solaris, HP/UX, AIX, and Linux). With those OS-specific needs satisfied, a layered driver may locate the external volume within the storage device by locating the external volume at a standard offset. Alternatively, the layered driver may locate the external volume within the storage device by encoding the offset within some block of the storage device that does have a known offset or by encoding that offset within some property of the disk (e.g., within a SCSI mode page).

[0061] In one embodiment, preferred, suggested, or mandatory names may be associated with storage devices. Naming a block device may allow applications and file systems to know what they are accessing. Moreover, naming a block device may allow the administrator of an environment to understand the relationship between the device that the file system is using and the virtual volume defined by the block virtualization. Block devices may be named by using software running on the host with the file system. That software might use symbolic links, alternate device files, or similar mechanisms to create an alias of a device node normally presented by the disk driver. Alternately, a local driver that sits above the disk driver may perform remapping to provide a local name space and devices that are coordinated with the external virtualization.

[0062] In one embodiment, a management communication layer may provide coordination of the naming function. In one embodiment, the basic properties of the external virtualized Logical Unit (e.g., through an extended SCSI mode page) may be extended to include a suggested or preferred name. In another embodiment, the name may be provided in a part of the emulated volume that resides outside the address space dedicated to the encapsulated volume.

[0063] The method set forth in Figure 5 may be performed a plurality of times, each for a different client computer and/or different operating system. As illustrated in Figure 4, a plurality of OS-specific emulated volumes 107 may be generated for a single volume 105.

[0064] In one embodiment, the storage volume may be moved from a host (e.g., a LAN-connected computer system) to the network (e.g., the SAN) prior to performing the method of Figure 5. In another embodiment, the method and system described herein may be implemented using host-based storage devices rather than off-host, SAN-based storage. In this embodiment, the method of Figure 5 may be performed to generate emulated storage volumes 107 for storage volumes 105 which are managed by a host computer system (typically on a LAN).

[0065] It is noted that the steps described above in conjunction with the descriptions of Figure 5 are numbered for discussion purposes only and that the steps may numbered differently in alternative embodiments.

[0066] In one embodiment, emulated storage volumes can be dynamic: volumes can be created, grown, shrunk, deleted, and snapshotted. In one embodiment, emulated storage volumes can be assigned and unassigned to systems. These operations may typically be performed synchronously and online.

[0067] With Logical Units, on the other hand, these various dynamic operations may carry undesirable overhead. The operations may be at least partially asynchronous, having unbounded completion times and ambiguous failures. On some operating systems, a system reboot may be required to complete some of these operations.

[0068] To reduce the overhead, one or more Logical Units may be created in advance of their use by client systems. The pre-provisioned Logical Units may contain the appropriate emulated metadata and may be pre-assigned to client systems. The presence of the emulated metadata may permit the storage stack on client systems to recognize these pre-provisioned Logical Units.

[0069] To associate an emulated storage volume with a client system, the volume may be assigned to a pre-provisioned Logical Unit. The assignment may increase the size of the Logical Unit to include the volume, and the assignment may adjust the emulated metadata as necessary to point to the volume and to adjust to the new volume and Logical Unit size.

[0070] In one embodiment, the storage stack (e.g., the disk driver) may be altered to recognize the new Logical Unit size and/or re-read the emulated metadata (e.g., to re-read the emulated OS partition table). In one embodiment, the pre-provisioned Logical Units may be given a size that matches the maximum Logical Unit size. Alternatively, the pre-provisioned Logical Units may be given a size that matches the maximum emulated volume size. In either case, the pre-provisioned Logical Unit would largely comprise unmapped blocks.

[0071] In one embodiment, an emulated volume may be mapped to multiple Logical Units to support larger volumes. In one embodiment, multiple emulated volumes may be mapped to one Logical Unit to reduce the number of pre-provisioned Logical Unit assignments.

[0072] Various embodiments may further include receiving, sending or storing instructions and/or data implemented in accordance with the foregoing description upon a carrier medium. Generally speaking, a carrier medium may include storage media or memory media such as magnetic or optical media, e.g., disk or CD-ROM, volatile or non-volatile media such as RAM (e.g. SDRAM, DDR SDRAM, RDRAM, SRAM, etc.), ROM, etc. as well as transmission media or signals such as electrical, electromagnetic, or digital signals, conveyed via a communication medium such as network and/or a wireless link.

[0073] The various methods as illustrated in the Figures and described herein represent exemplary embodiments of methods. The methods may be implemented in software, hardware, or a combination thereof. The order of method may be changed, and various elements may be added, reordered, combined, omitted, modified, etc.

[0074] Various modifications and changes may be made as would be obvious to a person skilled in the art having the benefit of this disclosure. It is intended that the invention embrace all such modifications and changes and, accordingly, the above description to be regarded in an illustrative rather than a restrictive sense.